Local Representations of Binding

Randy Pollack

LFCS, University of Edinburgh

Joint work with
James McKinna, Christian Urban, Arthur Charguéraud,
Brian Aydemir, Benjamin Pierce, Stephanie Weirich

Version of July 24, 2007

Local Representations

- Use syntactically distinct classes for (locally) bound variables vs (globally bound) “free” parameters.
  - The idea goes back to Gentzen and Prawitz.
- Locally named representation uses distinct species of names.
  - Alpha equivalence classes do not have canonical representatives.
  - McKinna–Pollack (1993) used this representation to formalize Pure Type System metatheory.
- Locally nameless representation uses names for parameters, and de Bruijn indices for locally bound variables.
  - Alpha equivalence classes have canonical representatives.
  - Mentioned by de Bruijn in his original paper.
  - Used by Huet in Constructive Engine.
  - Used for reasoning about binding by Andy Gordon (1994).

Local Representations

- Both are concrete representations
  - Close to informal usage.
  - “Anything true can be proved”
- Infrastructure for LNamed may be easier: no de Bruijn indices.
- LNameless better matches the concept of binding.
- There are a choice of technologies that make both local representations surprisingly convenient for significant metatheoretic reasoning.

Outline

Locally Named Representation

Raw Syntax

Simply Typed Lambda Calculus

Strengthened Induction Principle: McKinna–Pollack Style

Variations on McKinna–Pollack style

Nominal Isabelle Infers Strong Induction Principle

Reduction: Locally named seen to be unsatisfactory

Locally Nameless Representation

Substitution gets more complicated

The technology for strengthening elimination is the same

Conclusions

Outline

Locally Named Representation

- Raw Syntax
- Simply Typed Lambda Calculus
- Strengthened Induction Principle: McKinna–Pollack Style
- Variations on McKinna–Pollack style
- Nominal Isabelle Infers Strong Induction Principle
- Reduction: Locally named seen to be unsatisfactory

Locally Nameless Representation

- Substitution gets more complicated
- The technology for strengthening elimination is the same

Conclusions

Term Syntax

Names:

- A countable set, $\mathcal{V}$, of *variables*, ranged over by x, y, u, v.
- A countable set, $\mathcal{P}$, of *parameters*, ranged over by p, q, r.
- The only relations needed on $\mathcal{V}$ and $\mathcal{P}$ are decidable equality.
  - Nominal Isabelle provides types of *atoms* that behave this way.
  - Could have order relation on variables and/or parameters.

Terms:

- The syntax of pure λ-terms (ranged over by t, s, a, b):

$$t ::= x \mid p \mid t\ s \mid \lambda x.t$$

- In other settings, there may be other classes of parameters
  - e.g. type parameters and term parameters in $F_{<:}$,

  and other classes of variables and terms.

Freshness

- Define $p \sharp X$ means "p does not occur syntactically in X".
- We use $p \sharp X$ polymorphically for ...
  - p from any type of parameters
  - X from types of structures: terms, contexts, judgements, ...
- Each instance of $\sharp$ is easily defined by structural recursion.
- In nominal Isabelle, our $\sharp$ corresponds to nominal freshness (also written $\sharp$).
- Nominal Isabelle provides $\sharp$ polymorphic over classes of parameters and finitely supported structures for free.

Two Operations of “Substitution” (1)

- When going under a binder, a “hole” is created, i.e. a free variable.
- This operation fills such a hole:

$$
\begin{array}{lcl}
[s/y]x & = & \text{if } y = x \text{ then } s \text{ else } x \\
[s/y]q & = & q \\
[s/y](\lambda x.b) & = & \lambda x.(\text{if } y = x \text{ then } b \text{ else } [s/y]b) \\
[s/y](b_1\ b_2) & = & ([s/y]b_1)\,([s/y]b_2)
\end{array}
$$

- In the lambda case, this respects binding scope.
- However it does not prevent capture.
  - E.g. $[x/y]\lambda x.y = \lambda x.x$.
  - It will only be used in “safe” ways.

Two Operations of "Substitution" (2)

- Replacing a parameter by a term is entirely textual:

$$\begin{array}{lcl} [s/p]x & = & x \\ [s/p]q & = & \text{if } p = q \text{ then } s \text{ else } q \\ [s/p](\lambda x.b) & = & \lambda x.[s/p]b \\ [s/p](b_1\ b_2) & = & ([s/p]b_1)\,([s/p]b_2) \end{array}$$

- Both operations are defined by *structural* recursion.
- Both are deterministic: no choosing arbitrary names.
- Both have natural properties; e.g.
  - $[p/p]a = a$.
  - If $p \sharp a$ then $[s/p]a = a$.
- Neither prevents capture; they will only be used in "safe" ways.

Some Lemmas

Freshness

- $p \sharp ([s/n]t) \Rightarrow x \sharp t$
- $p \sharp (s, t) \Rightarrow p \sharp ([s/x]t)$

Substitution

- $p \sharp (s, t) \wedge [p/x]s = [p/x]t \quad \Rightarrow \quad s = t$
- $x \neq y \Rightarrow [q/x][p/y]s = [p/y][q/x]s$
- $p \sharp t \Rightarrow [u/p][p/y]t = [u/y]t$

All proved by structural induction.

Variable-Closed Terms

Since substitution operations allow capture, need a predicate meaning “no free variables”.

$$\frac{}{\mathit{vclosed}\ p} \qquad \frac{\mathit{vclosed}\ s \quad \mathit{vclosed}\ t}{\mathit{vclosed}\ (s\ t)} \qquad \frac{\mathit{vclosed}\ ([p/x]t)}{\mathit{vclosed}\ \lambda x.t}$$

- When going under a binder, substitute a parameter in the hole created.
  - The choice of p is arbitrary; we will have more to say.
- Every parameter is *vclosed* and no variable is *vclosed*.
  - When we induct over a *vclosed* derivation, there is no case for free variables!
- After some initial lemmas about raw terms, we always work with *vclosed* terms.
  - Use *vclosed* induction instead of term structural induction.

Variable-Closed Terms(2)

- Think of *vclosed* as a "weak typing judgement".
  - *vclosed* behave well for substitution, just as well-typed terms behave well for computation.
- $(vclosed\ s)$ is provably equivalent to " s has no free variables".
  - Thus *vclosed* is intuitively correct.
  - This fact not formally used.
- Example lemmas:

$$vclosed\ s \Rightarrow s = [p/n]s$$

$$p \neq q \wedge vclosed\ s \Rightarrow [s/p][q/n]t = [q/n][s/p]t$$

Remark: In intensional logics (like HOL) non-empty predicate *vclosed* can be made into a type.

- Andy Gordon (1994) does this in a slightly different context.
- I haven't experimented with this yet.

Simply Typed Lambda Calculus (STLC)

- Let $A, B, \ldots$ be *simple types* (implicational propositions).
- *Valid contexts* (Γ, Δ) are lists of uniquely labelled assumptions

$$\frac{\Gamma \text{ valid} \qquad p{:}A \in \Gamma}{\Gamma \vdash p : A} \qquad (\text{ELIM}) \; \frac{\Gamma \vdash b : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash b\, a : B}$$

$$(\text{INTRO}) \; \frac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \to B}$$

- When going under a binder, substitute a suitably fresh parameter in the hole created.
  - The choice of p is arbitrary; we will have more to say.
- Why no mention of *vclosed* ?
  - **lemma:** $\Gamma \vdash b : B \;\Rightarrow\; \mathit{vclosed}\; b$.

Simply Typed Lambda Calculus (2)

$$\frac{\Gamma \text{ valid} \qquad p{:}A \in \Gamma}{\Gamma \vdash p : A} \qquad (\text{ELIM})\ \frac{\Gamma \vdash b : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash b\,a : B}$$

$$(\text{INTRO})\ \frac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \to B}$$

- The side condition is needed in rule INTRO to prevent too many judgements being derivable:
  - if $p \in b$ then $p \in \lambda y.b$ in the conclusion, where p is not bound in the context Γ.
- Validity side conditions are *not* required in rules INTRO and ELIM because they follow from the premises.
- This definition of $\vdash$ is easily formalised in Coq, Isabelle/HOL, ...

Simply Typed Lambda Calculus (3)

What can we do with the definition of typing?

Using a technology which I will present shortly:

- Definitions and statements of lemmas are natural using names.
- All the expected judgements are derivable:
  - The set of derivable judgements is closed under alpha-conversion and renaming.
- The standard metatheory can be developed:
  - Weakening, substitution lemma, subject reduction ...
- We never need to define or reason about alpha-conversion.

Aside: Dependent Types

- McKinna/Pollack used this representation to formalize Pure Type Systems (PTS).
- One new point in typing rules:

$$\text{INTRO} \qquad \frac{\Gamma, p{:}A \vdash [p/x]M : [p/y]B \qquad p \sharp (M, B)}{\Gamma \vdash \lambda x{:}A.\, M : \Pi y{:}A.\, B}$$

  This is necessary for the following to be derivable in CC:

$$A{:}\star, P{:}A{\longrightarrow}\star \vdash \lambda x{:}A.\, \lambda x{:}Px.\, x : \Pi x{:}A.\, \Pi y{:}Px.\, Px \quad .$$

- Good intensional properties of this presentation.
  - **lemma:** $\Gamma \vdash M : B \;\Rightarrow\; \mathit{vclosed}\ M$.
  - Never need to define or reason about alpha-conversion.
  - Derivations closed under alpha-conversion and renaming.
  - Conversion rule only used for beta-conversion.

Weakening for STLC

- Define subcontext:

$$\Gamma \sqsubseteq \Delta \qquad \text{iff} \qquad \forall p, A \,.\, p{:}A \in \Gamma \;\Rightarrow\; p{:}A \in \Delta$$

Δ contains every assumption occurring in Γ.

Lemma (Weakening)

$$\Gamma \vdash a : A \;\wedge\; \Gamma \sqsubseteq \Delta \;\wedge\; \Delta \text{ valid} \;\Rightarrow\; \Delta \vdash a : A \,.$$

Remark (de Bruijn notation precludes natural statements)

- *If we use de Bruijn indexes for free (global) variables, neither the definition of $\sqsubseteq$ nor the statement of the lemma take the natural forms given above.*
- *Permuting the context requires lifting free indexes.*

Prove weakening

$$\Gamma \vdash a : A \;\wedge\; \Gamma \sqsubseteq \Delta \;\wedge\; \Delta \text{ valid} \;\Rightarrow\; \Delta \vdash a : A\,.$$

Proof: Attempt proof by induction on the derivation of $\Gamma \vdash a : A$

- Consider case for rule INTRO: $\dfrac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \rightarrow B}$
- By rule INTRO we need to show

$$\Delta, p{:}A \vdash [p/y]b : B \quad \text{for any } p \sharp b\,.$$

using IH:

$$\forall \Phi\,.\,(\Gamma, p_0{:}A \sqsubseteq \Phi \wedge \Phi \text{ valid}) \;\Rightarrow\; \Phi \vdash [p_0/y]b : B$$

for some particular $p_0 \sharp b$.

- It seems we want to instantiate Φ in IH with $\Delta, p_0{:}A$ ...
- ... but $\Delta, p_0{:}A$ may not be valid, as p_0 may occur in Δ.

Proof of weakening (contd)

- Since p_0 may not be fresh enough, we want to exchange it for a fresh parameter.
- Let $(p, q){\cdot}b$ mean *permute all occurrences of* p *and* q *in* b.
- As a lemma (equivariance), show

$$\Gamma \vdash a : A \;\Rightarrow\; \forall p\, q\,.\; (p, q){\cdot}\Gamma \vdash (p, q){\cdot}a : A\,. \tag{1}$$

  - This is easy to prove, but even better ...
  - nominal Isabelle defines polymorphic permutations and proves equivariance automatically.
- Now, pick $q \sharp (\Delta, b, \Gamma)$. It suffices to show

$$\Delta, q{:}A \vdash [q/y]b : B$$

which, by (1) and IH is difficult but possible.

We have a proof; what's the problem?

- We have to prove the equivariance property for every new judgement.
  - For some judgements, it isn't as easy as this example.
  - For some examples, Nominal Isabelle can't do it automatically.

But even if we use a meta-logic that proves equivariance uniformly ...

- ...we must still use name swapping explicitly (as in the *weakening* proof above) to handle each example where eigenvariable problems appear.
  - That is very heavy!

Better: we can package this swapping reasoning for each relation (typing, reduction, ...) once and for all.

- This technique from McKinna/Pollack (1993).

A more uniform solution to eigenvariable problems

The following judgements are equivalent:

- Arbitrary choice of p in INTRO: judgements may have infinitely many derivations:

$$\frac{\Gamma\ \text{valid} \qquad p{:}A \in \Gamma}{\Gamma \vdash p : A} \qquad (\text{ELIM})\ \frac{\Gamma \vdash b : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash b\,a : B}$$

$$(\text{INTRO})\ \frac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \to B}$$

- No arbitrary choices: judgements have at most one derivation:

$$\frac{\Gamma\ \text{valid} \qquad p{:}A \in \Gamma}{\Gamma \Vdash p : A} \qquad (\text{ELIM})\ \frac{\Gamma \Vdash b : A \to B \qquad \Gamma \Vdash a : A}{\Gamma \Vdash b\,a : B}$$

$$(\text{INTRO})\ \frac{\forall p\,.\, p \sharp \Gamma \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B}{\Gamma \Vdash \lambda y.b : A \to B}$$

A more uniform solution to eigenvariable problems

- $\vdash$ is the "official" relation, $\Vdash$ is an auxiliary notion.
- $\vdash$ is ordinary syntax: formalizable in Primitive Recursive Arithmetic.
- $\Vdash$, defined by generalized inductive definition, is not formalizable in PRA.
  - E.g. $\Vdash$ is not formalizable in Feferman's system FS_0.
- By induction on the derivation of $\Gamma \Vdash a : A$, it is trivial that

$$\Gamma \Vdash a : A \;\Rightarrow\; \Gamma \vdash a : A \;.$$

The other direction takes some work.

Why are we interested in this equivalence?

- It is easy to prove weakening:

$$\Gamma \Vdash a : A \;\wedge\; \Gamma \sqsubseteq \Delta \;\wedge\; \Delta \text{ valid} \;\Rightarrow\; \Delta \vdash a : A\,.$$

  hence, equivalently, weakening for $\vdash$.
- **Proof:** by induction on the derivation of $\Gamma \Vdash a : A$.
- Consider case for INTRO: $\dfrac{\forall p\,.\, p \sharp \Gamma \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B}{\Gamma \Vdash \lambda y.b : A \to B}$
- By rule INTRO we need to show

$$\Delta, p{:}A \vdash [p/y]b : B \quad \text{for some } p \sharp b\,.$$

  using IH:

$$\forall p\, \Phi\,.\, (p \sharp \Gamma \wedge \Gamma, p{:}A \sqsubseteq \Phi \wedge \Phi \text{ valid}) \;\Rightarrow\; \Phi \vdash [p/y]b : B\,.$$

- Select $p_0 \sharp (b, \Gamma, \Phi)$ and instantiate Φ in IH with $\Delta, p_0{:}A$.

Proof of the equivalence of $\vdash$ and $\Vdash$

Lemma $\Gamma \vdash a : A \;\Rightarrow\; \Gamma \Vdash a : A$.

- Proof by induction on the derivation of $\Gamma \vdash a : A$.
- Consider the case of rule INTRO.
- Any derivation of $\vdash$ will use a particular parameter, say p_0.
- The IH for this case is

$$\Gamma, p_0{:}A \Vdash [p_0/y]b : B \quad (p_0 \sharp b) \quad (\text{also } p_0 \sharp \Gamma)$$

  but to use the INTRO rule for $\Vdash$ we need the premise

$$\forall p \,.\, p \sharp \Gamma \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B$$

- How to reason from a particular parameter to all parameters?

Proof continued: $\Gamma \vdash a : A \;\Rightarrow\; \Gamma \Vdash a : A$

We solve the problem using swapping, as in the *weakening* proof.

- As a lemma, have equivariance of $\Vdash$:

$$\Gamma \Vdash a : A \;\Rightarrow\; \forall p\, q\, .\, (p, q)\cdot\Gamma \Vdash (p, q)\cdot a : A. \tag{2}$$

  Nominal Isabelle can't prove this automatically yet, but provides the lemmas about $\sharp$ and $(-,-)\cdot-$ that we need.

- We are trying to prove

$$\forall p\, .\, p \sharp \Gamma \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B\, .$$

  So pick $p \sharp \Gamma$. (Hence $p \sharp b$.)

- From IH and (2) have

$$(p, p_0)\cdot(\Gamma, p_0{:}A) \Vdash (p, p_0)\cdot([p_0/y]b) : B$$

  i.e. $\Gamma, p{:}A \Vdash [p/y]b : B$ as required.

Why are we interested in this equivalence?

- With this equivalence we can prove weakening by rule induction without name swapping.
- The equivalence of $\vdash$ and $\Vdash$ "packages" the eigenvariable reasoning that we need for all examples.
  - Introduction of $\vdash$ is easy: only need property for one fresh variable.
  - Elimination of $\Vdash$ is powerful: get the IH for all sufficiently fresh variables.
  - We use $\Vdash$ as a "derived induction principle" for $\vdash$
- Instead of using swapping arguments for every rule induction on every relation (typing, reduction, . . .) we only use it once for each relation.

Aside: Stronger inversion principles

- Rule INTRO $\dfrac{\Gamma, p{:}A \vdash [p/y]b : B \quad p \sharp b}{\Gamma \vdash \lambda y.b : A \to B}$

  gives rise (by induction) to an *inversion principle*:

  $$\Gamma \vdash \lambda y.b : T \;\Rightarrow\; \exists A, B, p\,.\,\Gamma, p{:}A \vdash [p/y]b : B \wedge p \sharp b \wedge T = A \to B\,.$$

- Rule INTRO $\dfrac{\forall p\,.\,p \sharp \Gamma \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B}{\Gamma \Vdash \lambda y.b : A \to B}$

  gives a stronger inversion principle (using $\vdash \Leftrightarrow \Vdash$):

  $$\Gamma \vdash \lambda y.b : T \;\Rightarrow\; \exists A, B\,.\,\forall p \sharp \Gamma\,.\,\Gamma, p{:}A \vdash [p/y]b : B \wedge p \sharp b \wedge T = A \to B\,.$$

All relations on terms get alternative definitions

- For example, *vclosed* ...

$$\frac{}{\mathit{vclosed}\ p} \qquad \frac{\mathit{vclosed}\ s \quad \mathit{vclosed}\ t}{\mathit{vclosed}\ (s\ t)} \qquad \frac{\mathit{vclosed}\ ([p/x]t)}{\mathit{vclosed}\ \lambda x.t}$$

...has a corresponding *Vclosed*

$$\frac{}{\mathit{Vclosed}\ p} \qquad \frac{\mathit{Vclosed}\ s \quad \mathit{Vclosed}\ t}{\mathit{Vclosed}\ (s\ t)} \qquad \frac{\forall p\ .\ \mathit{Vclosed}\ ([p/x]t)}{\mathit{Vclosed}\ \lambda x.t}$$

- $\mathit{vclosed}\ t \Leftrightarrow \mathit{Vclosed}\ t$

Do we need two relation forms, with an ad hoc equivalence proof, for every judgement we define?

Getting by with one relation form

You might wonder if we could use $\Vdash$ as the "official" typing relation, and avoid defining $\vdash$ at all.

- One objection is that $\Vdash$ is not really syntax.
- Another objection is that faithfullness of the representation to informal style is not obvious.
- Finally, the proofs don't all work!
  - Example: substitution lemma of STLC seems to be unprovable with only $\Vdash$:

    $$\Gamma_1, p{:}A, \Gamma_2 \Vdash b : B \;\Rightarrow\; \Gamma_1 \Vdash a : A \;\Rightarrow\; \Gamma_1, \Gamma_2 \Vdash [a/p]b : B$$

  - Induction is strong, but introduction of $\Vdash$ is too weak.

Cofinite Quantification: A different typing relation

- A better idea along the same lines: define a relation $\Vdash$ whose INTRO rule is

$$(\textsc{Intro})\ \frac{\forall p\,.\, p \sharp L \;\Rightarrow\; \Gamma, p{:}A \Vdash [p/y]b : B}{\Gamma \Vdash \lambda y.b : A \rightarrow B}$$

  where L is any (finite) list of parameters.
- L is existentially quantified in the premise of this rule.
  - Just as p is existentially quantified in the premise of

$$(\textsc{Intro})\ \frac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \rightarrow B}$$

- Cofinite quantification was used by Andy Gordon (1994).
- This approach strongly supported by Arthur Charguéraud *et.al.*
  - many interesting examples in Coq (e.g. POPLmark challenge).

Cofinite Quantification(2) (Charguéraud)

$$(\textsc{intro})\ \frac{\forall p\,.\; p \sharp \color{red}{L} \;\Rightarrow\; \Gamma, p{:}A \Vdash\!\vdash [p/y]b : B}{\Gamma \Vdash\!\vdash \lambda y.b : A \to B}$$

- $\Vdash\!\vdash$ lies "between" $\vdash$ and $\Vdash$: it is trivial to show that

$$\Vdash \;\Rightarrow\; \Vdash\!\vdash \;\Rightarrow\; \vdash$$

- $\Vdash\!\vdash$ has the same drawbacks as $\Vdash$ ...
  - ...almost all the proofs go through without referring to any other version of typing.
  - Decidability of typechecking doesn't go through directly:

$$\Gamma \text{ valid} \;\Rightarrow\; \mathit{vclosed}\ a \;\Rightarrow\; \Gamma \vdash a : A \;\vee\; \Gamma \nvdash a : A$$

- Enough proofs go through to show $\vdash \;\Rightarrow\; \Vdash\!\vdash$ without swapping.
  - Proves the equivalence $\vdash \Leftrightarrow \Vdash\!\vdash$ without swapping infrastructure.
- But it seems one can't prove $\Vdash\!\vdash \;\Rightarrow\; \Vdash$ without swapping.
  - A true statement that can't be proved in this style.

Don't need to state two relations at all (Xavier Leroy)

- I said you can prove $\vdash \Leftrightarrow \Vdash\!\vdash$ without swapping infrastructure.
- You can also prove

  lemma: $$\frac{\Gamma, p{:}A \Vdash\!\vdash [p/y]b : B \qquad p \sharp b}{\Gamma \Vdash\!\vdash \lambda y.b : A \to B}$$ is admissible in $\Vdash\!\vdash$

  without using swapping infrastructure.
- From this lemma it is trivial to show that $\vdash \Leftrightarrow \Vdash\!\vdash$ ...
- ...but there is no need to do so, or to define $\vdash$.
- If you take $\Vdash\!\vdash$ as the official definition of STLC (in spite of my objections above), this lemma can be used to prove properties of $\Vdash\!\vdash$, without defining other relations or developing swapping infrastructure.

- **BTW:** One can prove a similar lemma for $\Vdash$, but this needs swapping.

Nominal Isabelle: a strong induction principle

- Consider the typing relation $\vdash$ of STLC.
- Because $\vdash$ is variable-condition compatible, nominal Isabelle can infer a strengthened induction principle:

$$\dfrac{\begin{array}{l}\forall \Gamma, p, t, c\,.\;\; \mathit{valid}\,\Gamma \;\wedge\; (p{:}T) \in \Gamma \Rightarrow P\; c\; \Gamma\; (p)\; T \\[1ex] \forall \Gamma, t_1, t_2, T_1, T_2, c\,.\;\; \Gamma \vdash t_1 : T_1 \to T_2 \;\wedge\; \Gamma \vdash t_2 : T_1 \;\wedge \\ \quad (\forall d\,.\;\; P\; d\; \Gamma\; t_1\; (T_1 \to T_2)) \;\wedge\; (\forall d\,.\;\; P\; d\; \Gamma\; t_2\; T_1) \Rightarrow P\; c\; \Gamma\; (t_1\; t_2)\; T_2 \\[1ex] \forall p, \Gamma, t, T_1, T_2, v, c\,.\;\; p \,\sharp\, c \;\wedge\; p \,\sharp\, t \;\wedge\; \Gamma, p{:}T_1 \vdash [p/v]t : T_2 \;\wedge \\ \quad (\forall d\,.\;\; P\; d\; (\Gamma, p{:}T_1)\; ([p/v]t)\; T_2) \Rightarrow P\; c\; \Gamma\; (\lambda v.t)\; (T_1 \to T_2)\end{array}}{\Gamma \vdash t : T \Rightarrow P\; c\; \Gamma\; t\; T}$$

- “ c ” is a “freshness context” (finitely supported).
- Arbitrary name p chosen for λ case is fresh for c .

Variable-Condition Compatible

For details see (Urban, Berghofer and Norrish, 2007).
Roughly, an inductively defined relation is *vc-compatible* iff:

- The relation is equivariant.
  - For this to be true, every function and relation appearing in any premise or side condition must be equivariant.
- In each rule, the premises and side conditions imply that all arbitrarily chosen names are fresh for the conclusion.

$$\frac{\Gamma, p{:}A \vdash [p/y]b : B \qquad p \sharp b}{\Gamma \vdash \lambda y.b : A \to B}$$

  $p \sharp \Gamma$ by premise and $p \sharp b$ by side condition.
- Unfortunately, many relations are not vc-compatible, or require extra side conditions to become so.
- Don't know how to prove strong inversion principle from this IP.

Church–Rosser(1): Tait–Martin-Löf Parallel Reduction

$$\frac{}{p \gg p} \qquad \frac{s_1 \gg t_1 \qquad s_2 \gg t_2}{s_1\, s_2 \gg t_1\, t_2} \qquad \frac{[p/x]s \gg [p/y]t \qquad p \sharp (s,t)}{\lambda x.s \gg \lambda y.t}$$

$$\frac{[p/x]s_1 \gg [p/x]t_1 \qquad s_2 \gg t_2 \qquad p \sharp (s_1, t_1, s_2, t_2)}{(\lambda x.s_1)\, s_2 \gg [t_2/x]t_1}$$

- **lemma:** $s \gg t \;\Rightarrow\; vclosed\ s \;\&\; vclosed\ t$
- s_2, t_2 side condition needed to make rules vc-compatible.
  - Not needed for correctness!
- As above: fresh parameters fill hole when going under binder.
- With this definition (and the technology above) we easily prove that $\gg$ has the diamond property.
  - Hence, by strip lemma, $\gg^*$ has the diamond property.
- No need to define or reason about alpha-conversion.

Church–Rosser(2): Beta Reduction

Attempt to define beta-reduction:

$$(\beta) \qquad (\lambda x.b)\, s > [s/x]b$$

- Rule β is wrong, given our definition of $[s/x]b$, which allows capture.
  - E.g. the instance of β:

$$(\lambda x.\lambda y.x)\, y > [y/x]\lambda y.x = \lambda y.y$$

- What is the problem:
  - Only parameters may be free; variables must be bound.
  - "y" is not *vclosed*

Correct Beta Reduction

$$(\beta)\ \frac{\mathit{vclosed}\ \lambda x.b \qquad \mathit{vclosed}\ s}{(\lambda x.b)\, s > [s/x]b}$$

$$(\xi)\ \frac{[p/x]s > [p/y]t \qquad p \sharp (s,t)}{\lambda x.s > \lambda y.t}$$

$$\frac{s_1 > t \qquad \mathit{vclosed}\ s_2}{s_1\ s_2 > t\ s_2} \qquad \frac{s_2 > t \qquad \mathit{vclosed}\ s_1}{s_1\ s_2 > s_1\ t}$$

- In β, we must restrict to $(\mathit{vclosed}\ s)$ for safety.
  - There are no free variables in s that might be captured in $[s/x]b$.
- The other *vclosed* restrictions are for hygiene:
  - **lemma:** If $a > b$ then $\mathit{vclosed}\ a$ and $\mathit{vclosed}\ b$.

Church–Rosser (Contd.)

Now, can we finish the proof that $>^*$ has the diamond property?

- We know that $\gg^*$ has the diamond property.
- The standard proof now shows that $>^* = \gg^*$.
- Unfortunately this is false with my definitions:

$$\begin{array}{ccc} \lambda x.((\lambda x.x)\,x) & \xrightarrow{>} & \lambda x.x \\ {>}\big\downarrow & & \\ \lambda y.y & & \end{array}$$

  For $x \neq y$, this diagram cannot be closed.
- We have to state Church–Rosser up-to alpha-conversion.
- In locally named representation, alpha-equivalence shows through!
- We move to *locally nameless* representation.

Aside: How did we Formalize PTS without β-reduction?

- Define conversion using $\gg$ instead of $>$.
- $\gg$ has diamond property and Church–Rosser
- $\gg$ is much better behaved for coarse reasoning, such as subject reduction and Church–Rosser.

Outline

Locally Nameless Representation: Terms

- We use parameters, as before.
- Natural number de Bruijn indices serve for variables.
  - i, j, k, m, n range over indices.
- The syntax of pure λ-terms (ranged over by t, s, a, b):

$$t ::= i \mid p \mid t\ s \mid \lambda\, t$$

- As before, $p \sharp X$ defined for any structure X.
  - Provided automatically by nominal Isabelle.

Going under an abstraction: replacing a free index

- Correct for going under multiple abstractions simultaneously:

$$
\begin{array}{lcl}
[s/i]j & = & \text{if } j < i \text{ then } j \text{ (else if } i = j \text{ then } s \text{ else } (j-1)) \\
[s/i]q & = & q \\
[s/i](\lambda\, b) & = & \lambda[s/i+1]b \\
[s/i](b_1\; b_2) & = & ([s/i]b_1)\,([s/i]b_2)
\end{array}
$$

- An alternative that is much simpler to reason about, but correct only for going under a single binder:

$$
\begin{array}{lcl}
[s/i]j & = & \text{if } j = i \text{ then } s \text{ else } j \\
[s/i]q & = & q \\
[s/i](\lambda\, b) & = & \lambda[s/i+1]b \\
[s/i](b_1\; b_2) & = & ([s/i]b_1)\,([s/i]b_2)
\end{array}
$$

- We are only interested in $[s/0]t$, but proofs need to generalize to $[s/n]t$.

Replacing a parameter by a term

- Replacing a parameter by a term: just as for locally named.

$$\begin{array}{lcl} [s/p]i & = & i \\ [s/p]q & = & \text{if } p = q \text{ then } s \text{ else } q \\ [s/p](\lambda b) & = & \lambda([s/p]b) \\ [s/p](b_1\ b_2) & = & ([s/p]b_1)\,([s/p]b_2) \end{array}$$

- For locally nameless we need another function for replacing a parameter by an index:

$$\begin{array}{lcl} [k|p]i & = & i \\ [k|p]q & = & \text{if } p = q \text{ then } k \text{ else } q \\ [k|p](\lambda b) & = & \lambda([k{+}1|p]b) \\ [k|p](b_1\ b_2) & = & ([k|p]b_1)\,([k|p]b_2) \end{array}$$

- We are only interested in $[s|0]t$, but proofs need to generalize to $[s|n]t$.

The technology for strengthening elimination . . .

. . . is the same as for locally nameless as for locally named.

Outline

So which is the best representation?

For general purpose, large scale reasoning.

- Locally nameless: unique representation for each alpha-class.
- Locally named: alpha-equivalence shows through (e.g. Church–Rosser).
- I must go with locally nameless over locally named . . .
- although term infrastructure complicated by de Bruijn indexes.

Which technique for strengthening elimination?

1. McKinna–Pollack
   - Surely gets the job done.
   - Less heavy with nominal Isabelle infrastructure.
2. Cofinite quantification; swapping to prove equivalence.
   - If we're going to use swapping, why not go to strongest induction rule?
3. Cofinite quantification, Charguéraud's equivalence without swapping.
   - Very nice, works in many examples, but not yet well studied.
4. Cofinite quantification, Leroy's admissible rule without swapping.
   - Short and sweet, but I'm unhappy at not having the natural inductive definitions.
5. Is nominal Isabelle inferred induction rule ready yet?
   - I'm not sure (we only noticed it a month ago).
   - Biggest drawback is probably lack strengthened inversion.